Learning Word Subsumption Projections for the Russian Language

Dmitry Ustalov[1,2,*], *Alexander* Panchenko[3]

[1]Krasovskii Institute of Mathematics and Mechanics, 620990 Yekaterinburg, Russia
[2]Ural Federal University, 620002 Yekaterinburg, Russia
[3]Technische Universität Darmstadt, Computer Science Department, Language Technology Group, 64289 Darmstadt, Germany

Abstract. The semantic relations of hypernymy and hyponymy are widely used in various natural language processing tasks for modelling the subsumptions in common sense reasoning. Since the popularisation of the distributional semantics, a significant attention is paid to applying word embeddings for inducing the relations between words. In this paper, we show our preliminary results on adopting the projection learning technique for computing hypernyms from hyponyms using word embeddings. We also conduct a series of experiments on the Russian language and release the open source software for learning hyponym-hypernym projections using both CPUs and GPUs, implemented with the TensorFlow machine learning framework.

1 Introduction

In Linguistics, hyponymy denotes the asymmetric relationship between a generic term (hypernym) and a specific instance of this term (hyponym). These relations are similar to the relations between genus and species in Biology and called “subsumptions”. For instance, the word “cat” is a hyponym of the word “feline”. Traditionally, dictionaries of hypernyms and hyponyms are created manually by expert lexicographers or extracted automatically using lexico-syntactic patterns from a large collection of documents [1].

Since the inception of efficient methods for computing low dimensional word embeddings by Mikolov et al. [2], a significant attention has been paid to how distributional semantics can model relations of specific types, such as hypernyms or synonyms. One way to specify the type of relations between word vectors, investigated in this paper, is to induce a matrix such that multiplying on which a hyponym vector provides a hypernym vector. In particular, we investigate such an approach in the context of the Russian language.

In this paper, we will briefly review the related studies in Section 2 and describe the approach learning word subsumptions in Section 3, providing the open source implementation. We also conduct the performance study along with the quality evaluation in Section 4. Then, we discuss the obtained results in Section 5. Finally, we conclude with final remarks in Section 6.

* Corresponding author: dau@imm.uran.ru

2 Related Work

Currently, the most widely used method for detecting hypernyms and hyponyms is the Hearst patterns [1]. These lexical-syntactic patterns, e.g., "*Y such as* X_1 *and* X_2", have successfully found a substantial number of applications including ontology learning [3]. However, these patterns offer an inconvenient to work with the sparse representation of words which is being nowadays addressed using the word embeddings [2].

Fu et al. [4] proposed the projection learning approach to learning hypernyms for the Chinese language. This approach assumes learning the projection matrix such that multiplying on which a hyponym vector provides a hypernym vector. The learning problem has been posed as the linear regression problem that has been then numerically approximated using stochastic gradient descent. Also, the k-means clustering algorithm has been used to split the embeddings space to several subspaces to provide more flexibility to the model.

Levy et al. [5] observed the lexical memorization effect when using hyponym and hypernym embeddings for subsumption classification task. However, they conclude that it is still possible to learn "prototypical hypernyms", i.e., the word categories, due to the reported effect.

Kutuzov et al. [6] showed that the word embeddings can serve as informative language-independent semantic fingerprints when exploited in the problem of multilingual text clustering. In particular, the projection learning method similar to the one presented in our paper was used to translate words from Russian to Ukrainian, trained on a bilingual dictionary.

The method of Kutuzov et al. mentioned above stems from the original publication of Mikolov et al. [7], where projection learning was used to translate words from English to Spanish. More recently, Vulic et al. [8] presented a systematic study of four classes of methods for learning bilingual embeddings. The authors find approach based on linear projection, similar to the one we use in our method, to be most practical and efficient.

Vylomova et al. [9] evaluated several popular approaches for computing semantic relations and found that in word embeddings, vector subtraction generalises well to a broad range of relations, including over unseen lexical items.

Shwartz et al. developed an integrated method that combines the syntactic parsing features with word embeddings based on a long short-term memory network [10]. The resulting method called HypeNET has been implemented using the recurrent neural network that encodes the patterns with the embeddings.

3 Method

In the baseline setting proposed by Fu et al. [4], the projection matrix is obtained similarly to the linear regression problem, i.e., for the given row vectors $\mathbf{x}$ and $\mathbf{y}$ representing the hyponym and hypernym embeddings correspondingly, the $|\mathbf{x}| \times |\mathbf{y}|$ matrix Φ^* is numerically approximated:

$$\Phi^* = \arg\min_{\Phi} 1 / N \sum_{(\mathbf{x}, \mathbf{y})} \mathrm{dist}(\mathbf{x}\Phi, \mathbf{y}), \quad (1)$$

where N is the number of training examples and $\mathrm{dist}(\mathbf{x}\Phi, \mathbf{y})$ is the distance between a pair of row vectors $\mathbf{x}\Phi$ and $\mathbf{y}$. In the original method, the Euclidean distance (L^2 distance) is used. However, in distributional semantics, the cosine distance and similarity are the more widely used measures [2], so it is reasonable to study their performance. The distributed word representations tend to promote synonyms and other related words among the

hypernyms [11], which are of the primary interest. Thus, it seems also reasonable to provide the examples of undesired relations to refine the matrix being approximated.

3.1 Variations

Here, we propose three variations to the above-mentioned method: hyponymy penalization, synonymy penalization, and hypernymy promotion. Each variation consists of modifying the loss function by introducing the additional term weighted by the constant α or β that control the balance between two components of the loss function (in our experiments we used $\alpha = 0.01$ and $\beta = 0.3$. For preventing the difference from being negative, we use the absolute value.

3.1.1 Hyponymy Penalization

Our first variation is designed for enforcing the asymmetry of the projection matrix given the fact the subsumption is an asymmetric relation. Thus, applying the same transformation to the hypernym vector $\mathbf{x}\Phi$ as to the hyponym vector should not provide the initial hyponym vector $\mathbf{x}$.

$$\Phi^* = \arg\min_\Phi 1 / N |(1 - \alpha) \sum_{(\mathbf{x}, \mathbf{y})} \mathrm{dist}(\mathbf{x}\Phi, \mathbf{y}) - \alpha \sum_{\mathbf{x}} \mathrm{dist}(\mathbf{x}\Phi\Phi, \mathbf{x})| \quad (2)$$

3.1.2 Synonymy Penalization

Our second variation introduces the approach of negative sampling, i.e., explicitly providing the examples of synonyms $\mathbf{z}$ that penalizes the matrix to produce the vectors similar to them.

$$\Phi^* = \arg\min_\Phi 1 / N |(1 - \alpha) \sum_{(\mathbf{x}, \mathbf{y})} \mathrm{dist}(\mathbf{x}\Phi, \mathbf{y}) - \alpha \sum_{(\mathbf{x}, \mathbf{z})} \mathrm{dist}(\mathbf{x}\Phi\Phi, \mathbf{z})| \quad (3)$$

The main obstacle to realizing this loss function is the introduction of the $\mathbf{z}$ term representing a synonym of the given word $\mathbf{x}$, because certain words might have no synonyms. In such cases, we substitute $\mathbf{z}$ with $\mathbf{x}$, gracefully reducing to the previous variation. Otherwise, on each batch, we sample a random synonym of the given word.

3.1.3 Hypernymy Promotion

Our third variation is designed for promoting the projection matrix to produce hypernyms not just for the initial hyponym, but also for its randomly sampled synonym $\mathbf{z}$. This is motivated by the fact that in lexical ontologies the words are grouped into synsets (sets of synonyms) and the subsumptions are established between such synsets. So, both hyponym and its synonym are supposed to have the same hypernym.

$$\Phi^* = \arg\min_\Phi 1 / N [(1 - \beta) \sum_{(\mathbf{x}, \mathbf{y})} \mathrm{dist}(\mathbf{x}\Phi, \mathbf{y}) + \beta \sum_{(\mathbf{x}, \mathbf{z})} \mathrm{dist}(\mathbf{z}\Phi, \mathbf{y})] \quad (4)$$

In the case of no synonyms available, i.e., $\mathbf{x} = \mathbf{z}$, this variation gracefully reduces to the baseline setting.

3.2 Implementation

Instead of the linear regression used to approach this problem [4], our implementation is based on the single-layer perceptron developed using the TensorFlow open source

framework for machine learning [12] that supports both CPU and GPU computation of the numerical optimization procedures. Particularly, each input hyponym embedding x has been provided with an additional bias dimension. Thus, a vector $\mathbf{x}' = (1, \mathbf{x}_1, \ldots, \mathbf{x}_{|\mathbf{x}|})$ has been used instead of the original vector. Similarly, the projection matrix is now $|\mathbf{x}'| \times |\mathbf{y}|$. For minimizing the loss functions, we use the Adam stochastic optimization method [13]. We provide both implementations for L^2 and cosine distances for the loss functions, but our evaluation is focused only on the former due to the poor preliminary performance results of the latter.

4 Experiments

In our experiments, we use the following openly available language resources for Russian:

- pre-trained word embeddings in the form of 500-dimensional vectors computed using the skip-gram architecture [2] having the context window parameter as 10 words with the minimum word frequency of 5 (this model has been among the best ones in the RUSSE evaluation campaign [14]);
- a set of subsumption pairs obtained automatically using Hearst patterns from a large text corpus [14, 15];
- a set of subsumption pairs and synonyms derived from the Russian Wiktionary [16].

Particularly, as it has been suggested in [5], we split the train and test sets such that each contains a distinct vocabulary to avoid the lexical overfitting of the models. As the result, the training set contains 21 997 examples, the test set contains 10 811 examples. The test set contains only the examples from Wiktionary, while the training set is composed of other sources as well. We ran 14 000 training epochs; each passes a batch of 512 examples to the optimizer. The dimensions of the projection matrix are 501×500. At the initialization stage, we initialize the elements the projection matrix with N(0, 0.1). In the experiments, we study the performance of the loss functions operating with the L^2 distance along with the benefit of the clustering.

Since that the specificity of the relations differs in various regions of the embedding space, we employed the same clustering algorithm as described in [4, Section 3.3.2]. Initially, we estimated the number of clusters by maximizing the Silhouette score [17], but this approach led us to the suboptimal number of clusters $k = 2$. Instead, we evaluated all the values of $1 \le k \le 10$ to find the optimum on the test set. Each experiment has been run for five times to make it possible to assess the statistical significance of the results using the one-tailed t-test with the significance level of 0.025.

4.1 Quality Evaluation

In order to assess the quality of the model, we employed the following technique. For each subsumption pair $(\mathbf{x}, \mathbf{y})$ of hyponym $\mathbf{x}$ and the related hypernym $\mathbf{y}$ in the test set, we selected the projection matrix Φ_k^* assigned to the same cluster k as the given pair. Then, we compute ten nearest neighbours for the projected hypernym. The pair is considered matched if the word representing the gold hypernym $\mathbf{y}$ appears in the computed list of the nearest neighbours $\mathrm{NN}_{10}(\mathbf{x}\Phi_k^*)$. In order to obtain the integrated quality score, we average the matches across the test set:

$$\mathrm{A@10} = 1 / N \sum_{(\mathbf{x}, \mathbf{y})} \mathbb{1}(\mathrm{NN}_{10}(\mathbf{x}\Phi_k^*) \ni \mathbf{y}), \quad (5)$$

where N is the number of test examples and $\mathbb{1}(\ )$ is the indicator function. Intuitively, the A@10 measure is the probability of providing the correct hypernym among the ten nearest neighbours by projecting its related hyponym, which is previously unknown to the model.

Since the list of the nearest neighbours of the non-transformed hyponym vector may also contain hypernyms [11], it yields A@10 = 0.0877 on our test set.

4.2 Performance Study

Since that TensorFlow has been used for defining and executing the computation graph, we paid attention to the comparison of the CPU and GPU performance in our task. Therefore, for our experiments, we used the following computational resources available on a single machine:

- Intel Xeon E5-2620 v2 @ 2.10GHz (32 GB of RAM), denoted as CPU;
- NVIDIA Tesla K20Xm, 2866 cores (6 GB of VRAM), denoted as GPU.

5 Results and Discussion

According to the evaluation results in Table 1, both our variations implying penalizing the hyponymy and synonymy statistically significantly outperform the baseline in most settings. However, hypernymy promotion, inspired by lexical ontologies, showed the results worse than the baseline. Thus, we conclude that such a penalization can provide the system with the useful lexical information. Interestingly, no variation performed better than the baseline on $k = 6$ due to the inconsistent clustering.

Table 1. Quality evaluation according to A@10,
the best statistically significant result compared to the baseline in each setting is highlighted.

Model	***k*=1**	***k*=2**	***k*=3**	***k*=4**	***k*=5**	***k*=6**	***k*=7**	***k*=8**	***k*=9**	***k*=10**
Baseline	.1407	.2141	.2563	.2652	.3037	.3101	.3156	.3168	.3530	.3394
Pen. Hyponymy	**.1428**	.2161	.2591	.2700	.3100	.3080	.3255	.3268	.3655	**.3528**
Pen. Synonymy	.1427	**.2169**	**.2599**	**.2704**	**.3110**	.3093	**.3258**	**.3271**	**.3673**	.3526
Pro. Hypernymy	.1366	.2087	.2519	.2619	.3019	.3078	.3148	.3147	.3551	.3418

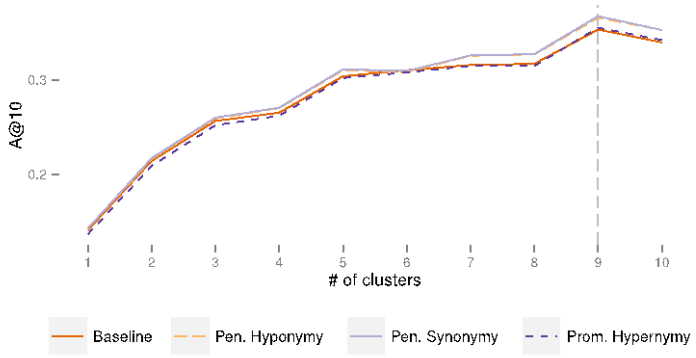

Fig. 1. Evaluation results according to the A@10 measure, the best result on all the variations is achieved on $k = 9$.

Increasing the number of clusters seems to be an efficient mean for increasing the capacity of the machine learning model. However, we found that the results stopped improving after $k = 9$, suggesting extending the training and test set sizes (Fig. 1). Since that the clustering reduces the number of the train items available per cluster, we had to use a relatively low batch size. To study the performance of the training procedure w.r.t. the

batch size, we run 1000 training epochs for the batch sizes of 512, 1024, 2048, 4096 and 8192. Table 2 shows the results of the performance study, confirming that under the present settings using a GPU makes the training process slower (Fig. 2) due also to the matrix size.

Table 2. Performance study: the total number of seconds spent per 1K training epochs on various batch sizes.

Model	Device	512	1024	2048	4096	8192
Baseline	CPU	1.52	2.05	3.18	7.61	14.38
Pen. Hyponymy	CPU	2.23	3.21	5.57	13.01	24.09
Pen. Synonymy	CPU	2.24	3.24	5.54	13.07	24.05
Pro. Hypernymy	CPU	1.77	2.64	4.59	10.87	20.47
Baseline	GPU	1.83	2.35	3.75	6.81	13.10
Pen. Hyponymy	GPU	2.39	2.95	4.39	8.00	13.92
Pen. Synonymy	GPU	2.57	3.22	4.99	8.87	15.98
Pro. Hypernymy	GPU	2.30	3.08	4.82	8.85	14.99

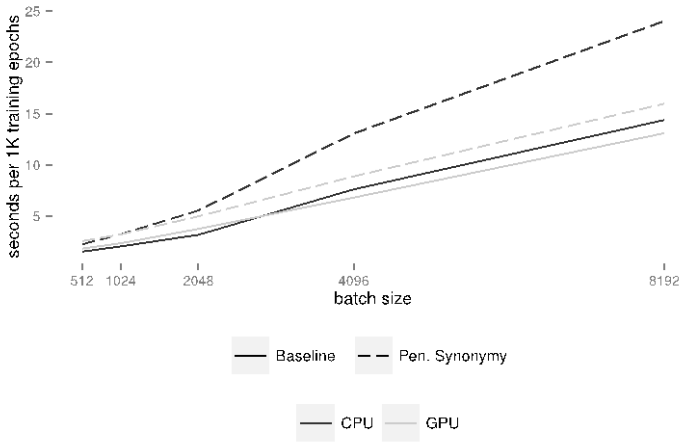

Fig. 2. Performance study of the baseline approach and the synonymy penalization approach involving negative sampling.

We also conducted a series of experiments with the cosine distance instead of the L^2 distance, but virtually in all the settings the A@10 measure was one and half times worse than using the L^2 distance, while the training process took ten times longer time. During these experiments, we removed the absolute value bars and replaced the negative distance terms in the equations (2) and (3) with the positive values of cosine similarity, making the loss function still non-negative.

6 Conclusion

In this study, we developed three models for learning word subsumptions and evaluated them on several resources for the Russian language. We also presented the open source software implementing the described approach, which is available under the terms of a libré license: https://github.com/dustalov/projlearn. Our datasets are available for other studies: http://ustalov.imm.uran.ru/pub/projlearn-ruwikt.tar.gz. To the best of our knowledge, this is

the first study dedicated to learning subsumptions using word embeddings for the Russian language.

In the further studies, we are interested in applying convolution layers for capturing high-level features of word embeddings, increasing the number of neural network layers, and using the learned matrices to construct semantic hierarchies. We also plan to conduct a crowdsourcing experiment to compare our results with the human judgements in order to weaken the dependency from the gold standard.

The reported study was funded by RFBR according to the research project No. 16-37-00354 мол_а. We are grateful to Nikolay Arefyev, Andrey Kutuzov, Andrey Krizhanovsky, Benjamin Milde and Alexander Bersenev for the fruitful discussions on the present study. Dmitry Ustalov was partially supported by the Deutscher Akademischer Austauschdienst (DAAD) scholarship. Alexander Panchenko was supported by the Deutsche Forschungsgemeinschaft (DFG) foundation under the project "JOIN-T: Joining Ontologies and Semantics Induced from Text".

References

1. M.A. Hearst. Automatic Acquisition of Hyponyms from Large Text Corpora. Proceedings of the 14th Conference on Computational Linguistics - Volume 2. COLING '92, Association for Computational Linguistics (1992) 539–545
2. T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean. Distributed Representations of Words and Phrases and their Compositionality. Advances in Neural Information Processing Systems 26. Curran Associates, Inc. (2013) 3111–3119
3. A. Panchenko, S. Faralli, E. Ruppert, S. Remus, H. Naets, C. Fairon, S.P. Ponzetto, C. Biemann. TAXI at SemEval-2016 Task 13 a Taxonomy Induction Method based on Lexico-Syntactic Patterns, Substrings and Focused Crawling. Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016). (2016) 1320–1327
4. R. Fu, J. Guo, B. Qin, W. Che, H. Wang, T. Liu. Learning Semantic Hierarchies via Word Embeddings. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1 Long Papers), Association for Computational Linguistics (2014) 1199–1209
5. O. Levy, S. Remus, C. Biemann, I. Dagan. Do Supervised Distributional Methods Really Learn Lexical Inference Relations? Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics Human Language Technologies, Denver, Colorado, Association for Computational Linguistics (2015) 970–976
6. A. Kutuzov, M. Kopotev, T. Sviridenko, L. Ivanova. Clustering Comparable Corpora of Russian and Ukrainian Academic Texts Word Embeddings and Semantic Fingerprints. Proceedings of the Ninth Workshop on Building and Using Comparable Corpora. (2016) 3–10
7. T. Mikolov, Q.V. Le, I. Sutskever. Exploiting Similarities among Languages for Machine Translation. CoRR abs/1309.4168 (2013)
8. I. Vulić, A. Korhonen. On the Role of Seed Lexicons in Learning Bilingual Word Embeddings. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1 Long Papers), Berlin, Germany, Association for Computational Linguistics (2016) 247–257
9. E. Vylomova, L. Rimell, T. Cohn, T. Baldwin. Take and Took, Gaggle and Goose, Book and Read Evaluating the Utility of Vector Differences for Lexical Relation Learning. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1 Long Papers), Berlin, Germany, Association for Computational Linguistics (2016) 1671–1682

10. V. Shwartz, Y. Goldberg, I. Dagan. Improving Hypernymy Detection with an Integrated Path-based and Distributional Method. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1 Long Papers), Berlin, Germany, Association for Computational Linguistics (2016) 2389–2398
11. A. Handler. An empirical study of semantic similarity in WordNet and Word2Vec. Master's thesis, University of New Orleans (2014)
12. M. Abadi et al. TensorFlow Large-Scale Machine Learning on Heterogeneous Distributed Systems. CoRR abs/1603.04467 (2016)
13. D.P. Kingma, J. Ba. Adam A Method for Stochastic Optimization. CoRR abs/1412.6980 (2014)
14. N. Arefyev, A. Panchenko, A. Lukanin, O. Lesota, P. Romanov. Evaluating Three Corpus-Based Semantic Similarity Systems for Russian. Computational Linguistics and Intellectual Technologies Papers from the Annual International Conference "Dialogue", Volume 2 of 2. Papers from special sessions, Moscow, RGGU (2015) 106–118
15. A. Panchenko, O. Morozova, H. Naets. A Semantic Similarity Measure Based on Lexico-Syntactic Patterns. Proceedings of KONVENS 2012, ÖGAI (2012) 174–178
16. A.A. Krizhanovsky, A.V. Smirnov. An approach to automated construction of a general-purpose lexical ontology based on Wiktionary. Journal of Computer and Systems Sciences International 52(2) (2013) 215–225
17. P.J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics 20 (1987) 53–65